Integrating Graphic Generalisation Methods in a System for Real-Time Maps

Lars Harrie

GIS-centre, Lund University
Sölvegatan 12, 223 62 Lund, Sweden
lars.harrie@lantm.lth.se

Abstract.
Internet and mobile technologies have made it possible to distribute data from a cartographic database to a computer or a mobile device. This technology is today used by several commercial and research map services. Most of these services deliver pre-processed maps, but there is a tendency to introduce real-time methods to tailor the maps for certain applications. This paper describes a study of integrating graphic generalisation methods (based on optimisation techniques) into a real-time map services. The generalisation methods as such are not new; the study is mainly aimed at issues of integrating the methods into a real-time system. Key issues here are where in the system architecture to implement the generalisation methods, how to structure the data (mainly establishing topological relationships) as required by the generalisation methods, and the processing time for the required processes. These issues are investigated in a feasibility study; the study shows promising result but also address the need of more thorough case studies.

Keywords: cartographic generalisation, data structure, optimisation, real-time maps, OGC

1 Introduction

Maps are important tools for visualising geographic locations. Traditionally, printed paper maps have been used, but technological developments have made possible the production of digital maps. Furthermore, the introduction of the Internet and mobile technology has made it possible for a mobile device to communicate with remote databases. This way cartographic data from a remote database, distributed in real-time, can be displayed locally as maps. In this paper we denote this kind of maps as *real-time maps*.

To create good real-time maps the cartographic data often requires to be generalised before it is distributed to the user. This generalisation could either be performed as a pre-process or in real-time. If the generalisation is run as a pre-process batch and interactive generalisation could be used. In this study we concentrate on generalisation performed on demand in real time. Recently, several studies have been dedicated to real-time generalisation applications (Lehto and Kilpeläinen, 2000; van Kreveld, 2001; Jones et al., 2000; Cecconi, 2003). Few studies, however, have been performed to include more complex generalisation routines in real-time environments.

The aim of this study is to use graphic generalisation routines to create real-time maps. The graphic generalisation is performed by optimisation techniques. The idea of optimisation techniques in cartographic generalisation is not new (see e.g. Burghardt and Meier, 1997; Ware and Jones, 1998; Harrie, 1999; Højholt, 2000; Sester, 2000; Bader, 2001; Harrie and Sarjakoski, 2002). This study will not develop these theories any further; instead it concentrates on the requirements for using this type of techniques in a system architecture for real-time maps. More specifically, the aim of this study is to adjust and extend previous implementation of simultaneous graphic generalisation (as presented in Harrie and Sarjakoski, 2002) to be used in a system architecture for real-time maps developed in the EEC-project GiMoDig (see Lehto, 2003; Sarjakoski and Lehto, 2003; GiMoDig, 2005).

The paper starts with a summary of the optimization technique simultaneous graphic generalisation. Then follows a summary of a system architecture for real-time maps. Section 4 describes a prototype system; this system is evaluated in a case study as described in Section 5. The paper concludes with a discussion.

2 Simultaneous graphic generalisation

This section summarizes the theory behind simultaneous graphic generalisation. For a detailed description see Harrie and Sarjakoski (2002).

The method simultaneous graphic generalisation is only concerned with the graphic part of generalisation (i.e. graphic changes of the symbolisation to make the map more readable, roughly corresponding to the generalisation operators simplification, smoothing, displacement and exaggeration); the method does not include any model generalisation (generalisation due to changes in the conceptual model). In several generalisation problems both model and graphic generalisation are required. However, several parts in the model generalisation (e.g. typification) are difficult to define analytically and, hence, difficult to implement in a fully automatic system. A solution to avoid these hard parts of the generalisation transformation in real time is to use a multiple representation database (Cecconi, 2003; Hampe et al., 2004).

Simultaneous graphic generalisation aims at computing the optimal solution according to a set of constraints. The following constraints are used:

- Displacement: Spatial conflicts are not allowed.
- Simplification: Line and area objects should not contain more points than necessary to represent their characteristics. The simplification constraints force an *unnecessary point* to lie on the straight line between the two neighbouring points.
- Smoothing: Line and area objects should not be too angular.
- Exaggeration: Objects, and features within objects, should be large enough to be clearly visible.
- Curvature and Segment length: The characteristics of line and area objects must be maintained.
- Stiffness: The internal geometry of some objects must be invariant.
- Crossing: The angle between line objects in junctions must not change.
- Exaggeration: The shape of some objects must be maintained.
- Movement: Points should not move.
- Movement direction: Points on a line should not move in any direction across the line.

One key issue is to find analytical expressions for the constraints. In simultaneous graphic generalisation the number of points is invariant, which enables the formulation of the constraints on point movements. For the sake of computational simplicity, we restrict ourselves to linear equations; that is, all the constraints are of the form:

$$const_{x1} \cdot \Delta x_1 + const_{y1} \cdot \Delta y_1 + \ldots + const_{xn} \cdot \Delta x_n + const_{yn} \cdot \Delta y_n = const_{obs} \tag{1}$$

where

$\Delta x_i, \Delta y_i$ are point movements,

$const_{xx}$ are constant values, and

n is the total number of points.

All the constraints together constitute an equation system in which the point movements are the unknowns. In matrix form this equation system can be written as:

$$\mathbf{Ax} = \mathbf{l} + \mathbf{v} \tag{2}$$

where

A is the design matrix,

x	is a vector containing the unknown point movements,
l	is the observation vector (containing the right-hand side of Equation (1)), and
v	is the residual vector.

The residual vector has to be introduced since the Equation system (2) is over-determined (i.e., more constraints than unknowns). The method guarantees that either a movement or a simplification constraint is set up for each *x*- and *y*-coordinate, and normally there are constraints set up for preserving characteristics or improving legibility. That is, there are always at least as many constraints as unknowns, and in realistic applications there are about twice as many constraints as unknowns.

The "best solution" of Equation system (2) is the one that agrees as far as possible with the constraints, i.e. we face a minimisation problem of a function of the residual vector (**v** in Equation system (2)). To solve the equation system the least-squares method is used, which minimises a weighted l_2 norm:

$$\mathbf{v}^{\mathrm{T}}\mathbf{P}\,\mathbf{v} \tag{3}$$

where

P	is the weighting matrix, and
$\mathbf{v}^{\mathrm{T}}$	is the residual vector transposed.

The least-squares solution of the unknown point movements (stored in vector **x**) is given by:

$$(\mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{A})\cdot\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{P}\,\mathbf{l}\ . \tag{4}$$

Finally, the map is generalised by adding the computed point movements to the original coordinates.

The whole process of simultaneous graphic generalisation can be viewed as follows. In the initial state, some constraints are severely violated (e.g. displacement and simplification) while other constraints are not violated at all (e.g. movement and stiffness). The generalisation process then distributes the violations more evenly over all the constraints; the degree to which the violations are spread is dependent on the weights stored in matrix **P**. A thorough discussion about these weights is given in Harrie (2003).

2.1 Computational aspects of simultaneous graphic generalisation

There are two main tasks that are computationally demanding in simultaneous graphic generalisation: solving the Equation system 4 and identifying the spatial relationships between the cartographic objects. Sarjakoski and Kilpeläinen (1999) proposed the use of conjugate gradient method to solve the Equation system 4. Conjugate gradient method is an iterative method that is storage and computationally efficient for sparse equation system (as is the case for Equation system 4). The spatial relationships are derived from constrained Delaunay triangulation (see Chew, 1989 for details, similar methods to identify spatial relationships are also used by Ruas and Plazanet, 1996; Ware and Jones, 1998; Højholt, 2000; Sester, 2000). Delaunay triangles can be computed in O(*n* log *n*) expected time, where *n* is the number of points (cf. de Berg et al., 1997).

Practical performance tests in Harrie and Sarjakoski (2002) indicated an expected computational complexity of O(*n* log *n*) time of simultaneous graphic generalisation. However, this value is much dependent on implementation solutions, type of cartographic data and parameter setting.

3 A system architecture for real-time maps

In theory, generalisation routines could be implemented in the client. However, this would require that the client have access to cartographic data (and not only a graphic representation of the data) and high process / programming capabilities. Currently, an implementation of complex generalisation routines in a client is not a realistic alternative. Therefore, these routines are required to be implemented in a service that could be reached by the client; hence, a kind of system architecture for real-time map applications is required. This system architecture could be based on Open Geospatial consortium (OGC) specification for cartographic web services as e.g. Web Map Service (WMS, 2005) and Web Feature Service (WFS, 2005).

The study described in this paper is part of the EC project GiMoDig (Geospatial info-mobility service by real-time data integration and generalisation; GiMoDig, 2005); which is a project that aims at establishing methods for distributing cartographic data from core databases at national mapping agencies to mobile devices (mainly following the OGC standards). Figure 1 is a simplified overview of the GiMoDig system architecture (see Lehto, 2003 or Sarjakoski and Lehto, 2003 for details). The client makes a WMS request for a map. The request is first transformed to an extended WFS request. This extended WFS request contains, besides the traditional WFS parameters, information about how to perform generalisation and integration (as described in Sarjakoski et al., 2005). The data processing layer then forward a WFS request to the data layer. The response to this request is cartographic data in GML format. The cartographic data are sent to the data processing layer. In this layer generalisation of the cartographic data is performed. The data processing layer then sends the generalised cartographic data to the portal layer. Finally, in the portal layer, the GML data is translated into, for example, an SVG or JPEG image for display at the client.

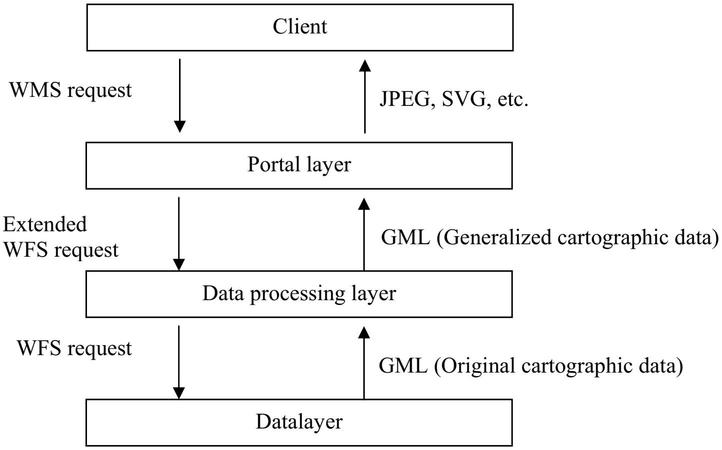

Figure 1. A simplified version of the GiMoDig system architecture.

The generalisation routines should be implemented in the data processing layer (for details see Sester et al., 2004). Figure 2 shows the workflow for this layer (which is implemented by Tommi Koivula and Lassi Lehto, Finnish Geodetic Institute). There are two programming environments for the generalisation: XSLT and Java/JTS. XSLT is a standard XML application for transforming one XML document into another (see e.g. Harold and Means, 2001). JTS (JTS Topology Suite) is an open source Java package from Vivid Solutions (2005) that conforms to the *Simple Features Specification for SQL* (OGC, 2005) and contains robust implementations of the most fundamental spatial algorithms in 2D. XSLT could be used for fast generalisation that does not require relationship between objects (see Lehto and Kilpeläinen, 2000, 2001a, 2001b). JTS is not that computationally fast; on the other hand the JTS environment provides tools for handling complex relationships between objects in the generalisation process (see Harrie and Johansson, 2003; Hampe et al., 2004; Harrie et al., 2004; Zhang and Harrie, 2005; Stigmar, 2005).

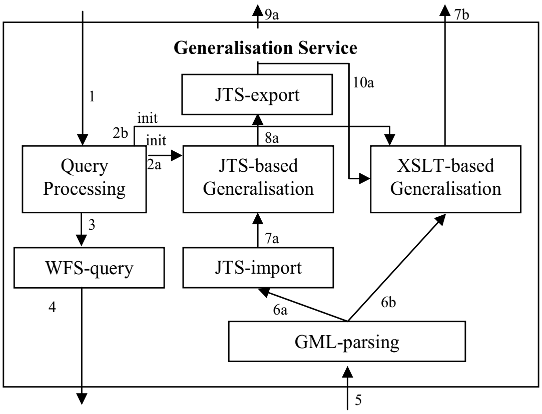

Figure 2. Generalisation Service in GiMoDig, internal workflow (Lehto, 2003).

4 Requirement of topological relationships

To properly define the constraints in a generalisation simultaneous graphic generalisation the topological relationships between the cartographic objects must be known. The reason is that the routine must be able to distinguish between spatial conflicts between objects (their symbols interfere with each other) and objects that are connected or intersect (for these objects there are no spatial conflict even though the symbols may overlap). Furthermore, the topological relationships are required to check that the graphic generalisation routine does not introduce any topological errors in the data. There are three alternatives to have access to topological relationships in the generalisation process (anticipating that the topological relationships are stored explicitly in the data layer):

1) Perform generalisation in the data layer.
2) Transport the topological relationships from the data layer to the layers above.
3) Compute the topological relationships in real-time (in the layer where the generalisation is performed).

In our study we would like to integrate the prototype into a system architecture similar to the one in Figure 1; then alternative 1 is not really a choice. But, principally, another system architecture could be used where the data layer and the data processing layer would be one single unit. The advantage then would be the access to topological relationships for e.g. generalisation processes. However, such a solution would not allow that the data layer consist of a cascading WFS solution (i.e., there are several WFS servers that provides data to a single output WFS server) (this solution is e.g. implemented in the GiMoDig project, see Lehto, 2003).

There is a lack of good tools for alternative 2. Most WFS servers provide GML2 data; this format does not support topological relationships. The new GML3 format gives such support (GML, 2005). But at the time of writing there are few (if any available?) WFS servers that actually supports GML3 and few (if any available?) parsers that could code GML3 data into topological data in e.g. a geometrical Java library.

That is, we found that alternative 3 is currently most interesting for us. This alternative is implemented in a prototype system and tested in a feasibility study (as described in Sections 5 and 6).

5 Prototype system

This section describes how the implementation of simultaneous graphic generalisation can be adjusted and extended to fit into GiMoDig system architecture (as described in Section 3). It should be noted that a complete integration into the GiMoDig system is not yet performed but the prototype is written according to the Java interfaces developed for the general workflow of the processing layer (cf. Figures 1 and 2). This implies that it would not be too much work to perform a complete integration.

5.1 Prototype structure

The implementation of simultaneous graphic generalisation in Harrie and Sarjakoski (2002) was written in c and c++. The program utilises c/c++ packages for Delaunay Triangulation (written by Shewchuk, 1996) and conjugate gradient method (written by Tapani Sarjakoski, Finnish Geodetic Institute). Furthermore, the program communicates with the object-oriented map production system LAMPS2 (Laser-Scan, 2004) via ASCII files. The program requires that the cartographic data is stored in a topological data structure in the LAMPS2 environment and the topological relationships are exported to the c/c++ program.

The data processing layer in the GiMoDig system architecture (Figure 2) is in a Java environment. To utilise the c/c++ program in this environment Java native interface (Gordon, 1998) was used. The division of the tasks between the Java and c/c++ parts of the prototype follows Figure 3. The reason for this division of the tasks was the wish to use as much of the original implementation of simultaneous graphic generalisation (written in c/c++) and a judgement that it would be easier to establish the topological relationships in Java.

With reference to Figure 3 the prototype works as follows:

1) Data is requested from a WFS-server.
2) GML data is parsed into Java objects. The parsing is performed by a package from JUMP (extended in the GiMoDig project, see Sester et al., 2004).
3) The topological relationships between the cartographic objects are identified. A call is performed to the c-program for simultaneous graphic generalisation using Java native interface. The function call passes data about the objects (coordinates, topological relationships, etc.) and how they should be treated in the generalisation process.
4) Simultaneous graphic generalisation is performed. The result from this process (the point movements in vector $\mathbf{x}$ in Equation 2) are returned to the Java environment.
5) The current objects are generalised by adding the point movements to their coordinate values.
6) The result is visualised in the JUMP graphical user interface.

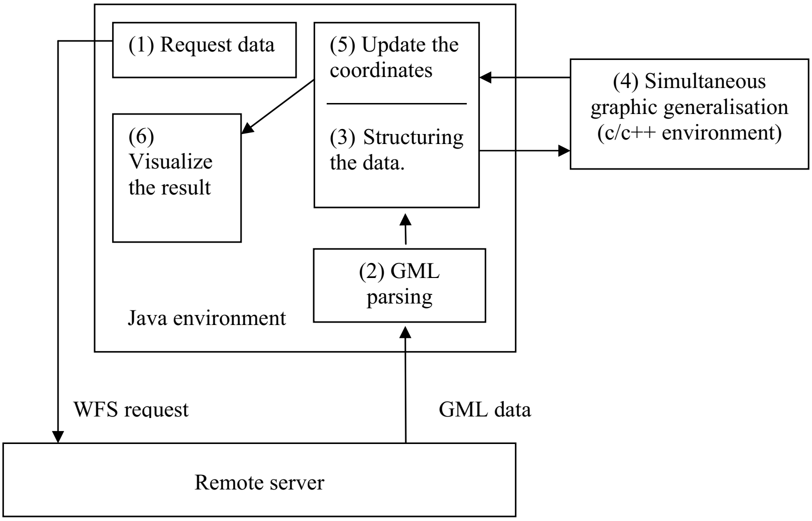

Figure 3. The structure of the prototype

5.2 Establishing topological relationships

Since we were using Java/JTS for other computation in the data processing layer we chose this environment also for identifying the topological relationships. In the prototype we used JTS to structure the data in real-time. In short, the routine works as follows:

1) Convert all polygon objects into closed line objects.
2) Put all point and line objects in a list.
3) Check if two first geometries in the list *intersect* and/or *meet*.
    a) If two objects *meet* then this information must be stored (this information is later used in the generalisation process to guarantee that the topological relationships are not lost).
    b) If two objects *intersect*. Divide the objects so that they follow the rules for a link-node data structure. Remove the original objects from the list and append the new objects to the list.
4) Go back to point three until all objects are checked.

In practice there are several problems regarding this structuring process. The main problem is the rounding errors (which occurs even though JTS uses "robust" algorithms). These rounding errors entail that you cannot always use the JTS topological functions (which require perfect data) but instead use an approach where e.g. links sufficiently close (i.e., closer than a threshold value) are defined to *meet*.

6 Feasibility study

The feasibility study shown here is the very first test of the program. The cartographic generalisation problem and the generalisation result is not the best. However, the main issue here is the performance of the program, and specially the relationship between the processing time of the different parts of the generalisation process.

6.1 Cartographic result

The feasibility study was performed using a web feature service at the Finnish Geodetic Institute (see Lehto, 2003). Only ten objects were used (cf. Figures 4 and 5):

- 2 minor road objects (the data is not perfect; this single link was actually divided into two separate links),
- 3 major road objects (one for each link), and
- 5 building objects.

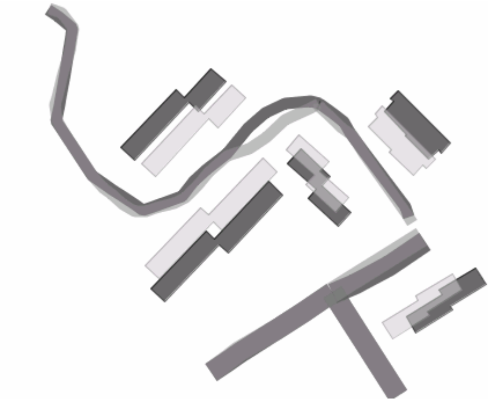

Figure 4. Result of the generalisation process. Original map in grey and generalised map in black. Cartographic data from National Land Survey of Finland

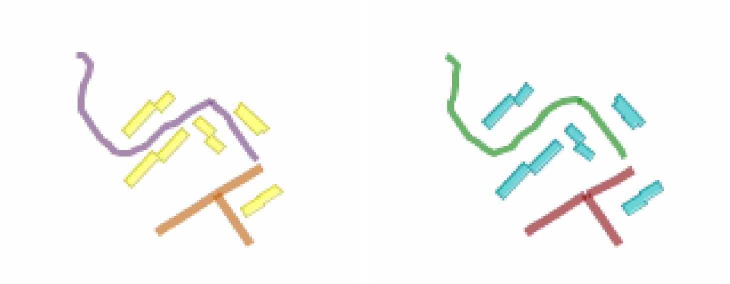

Figure 5. (a) The original map; (b) The generalised map. Cartographic data from National Land Survey of Finland

6.2 Performance result

The performance result is given in Table 1. The performance is divided into three different phases:

- retrieving data (steps 1 and 2 in section 5.1),
- structuring data (step 3 in section 5.1), and
- generalising data (steps 4 and 5 in section 5.1).

The same data as in Figures 4 and 5 were used and the computations were performed in a PC with a Pentium III processor.

Table 1: CPU-time for different phases in the prototype.

	Retrieving data (ms)	Structuring data (ms)	Generalising data (ms)
Test 1	1182	331	881
Test 2	1272	330	891

The data used in the test was comparatively easy to structure. A couple of objects *meet*, but no object *intersect*. Tests with fictitious geometric data indicates that it take longer times if some of the objects *intersect* (which is quite naturally).

7 Discussion

The main aim of this study is to evaluate the use of the optimisation routine simultaneous graphic generalisation in a real-time environment. The two main issues are:

- development of a method for establishing topological relationships in real-time, and
- computational performance of simultaneous graphic generalisation.

The cartographic aspects were not really studied. However, simultaneous graphic generalisation has shown good cartographic performance in previous studies (Harrie and Sarjakoski, 2002; Harrie, 2003) and this property does not change if the methods are integrated in a new environment. That is, the success of simultaneous graphic generalisation is mainly dependent on the two issues above.

In the prototype we integrated a quite simple approach for establishing the topological relationships (due to limited time for programming). It is possible to make more computationally efficient implementations than in the prototype (using e.g. plane sweep algorithms, cf. de Berg et al., 1997). But still the structuring of the data was, compared to the generalisation processes, not the main computational problem. More efficient structuring routines (and perhaps also tools for topology handling using GML3) will reduce the problem of establishing the topological relationships in the generalisation process environment.

The computational performance of simultaneous graphic generalisation might be somewhat problematic (for the computers of today). The feasibility study shows that even a small amount of data takes a quite a long processing time. And there are no well-known methods (at least not for the author) of how to improve the computational complexity.

To conclude, the use of simultaneous graphic generalisation for real-time maps still has to be verified. A larger case study should be performed with more realistic data sets and realistic computer resources (modern server computers).

Acknowledgements

The research described in this paper is part of the GiMoDig project, IST-2000-30090, which is funded by the European Union via the Information Society Technologies (IST) programme. The author would like to thank the colleagues in the GiMoDig project and Qingnian Zhang for their cooperation, especially Tommi Koivula and Lassi Lehto for kindly proving Java code and Tapani Sarjakoski for providing routines for conjugate-gradient method. Special thanks to Mikael Johansson for help with integrating the Java and C programs and to participants in the ICA/EusroSDR Workshop, Leicester, UK, August 2004, for valuable comments on this topic.

References

Bader, M., 2001. *Energy Minimization Methods for Feature Displacement in Map Generalization*, Doctorate Thesis, Geographic Information System Division, Department of Geography, University of Zurich, Switzerland.

Burghardt, D., and Meier, S., 1997. Cartographic Displacement Using the Snakes Concept. In Foerstner, W., and Pluemer, L. (eds), *Semantic Modeling for the Acquisition of Topografic Information from Images and Maps*, Birkhaeuser Verlag, pp. 59-71.

Cecconi, A., 2003. Integration of Cartographic Generalization and Multi-Scale Databases for Enhanced Web Mapping. PhD Thesis, University Zurich, Switzerland.

Chew, L. P., 1989. Constrained Delaunay Triangulations. *Algorithmica*, Vol. 4, pp. 97-108.

de Berg, M., M. van Kreveld, M. Overmars, and O. Schwarzkopf, 1997. *Computational Geometry: Algorithms and Applications*, Springer.

GiMoDig, 2005. *Geospatial info-mobility service by real-time data-integration and generalization*, http://gimodig.fgi.fi/ (accessed 2005-02-09).

GML, 2005. *Geographic Markup Language,* http://www.opengeospatial.org/docs/02-023r4.pdf (accessed 2005-02-09).

Gordon, R., 1998. *Essential JNI – Java Native Interface*. Prentice-Hall.

Hampe, M., Sester, M. and Harrie, L., 2004. Multiscale Databases to support visualisation on mobile devices. Proceedings of the XXth ISPRS Congress, July12-23, 2004, Istanbul, Turkey, *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXV(B4:IV), pp. 135-140.

Harold, E. R., and Means, W. S., 2001. *XML – in a Nutshell*. O'Reilly.

Harrie, L. 1999. The Constraint Method for Solving Spatial Conflicts in Cartographic Generalization. *Cartography and Geographic Information Science*, Vol. 26, No. 1, pp. 55-69.

Harrie, L., and Sarjakoski, T., 2002. Simultaneous Graphic Generalization of Vector Data Sets, *GeoInformatica,* Vol. 6, No. 3, pp. 233-261.

Harrie, L., 2003. Weight-Setting and Quality Assessment in Simultaneous Graphic Generalisation. *The Cartographic Journal*, Vol. 40, No.3, pp. 221-233.

Harrie, L. and Johansson, M., 2003, Real-time data generalization and integration using Java. *Geoforum Perspektiv*, Februar 2003, pp.29-34.

Harrie, L., Stigmar, H., Koivula, T., and Lehto, L., 2004. An Algorithm for Icon Placement on a Real-Time Map. In Fisher, P. (ed), *Development in Spatial Data Handling,* Springer, pp. 493-507.

Højholt, P., 2000. Solving Space Conflicts in Map Generalization: Using a Finite Element Method. *Cartography and Geographic Information Science*, Vol. 27, No. 1, pp. 65-73.

Jones, C. B., Abdelmoty, A. I., Lonergan, M. E., van der Poorten, P., and Zhou, S., 2000. Multi-Scale Spatial Database Design for Online Generalisation. *Proceedings of the 9th Spatial Data Handling Symposium*, Beijing, pp. 7b.34-7b.44.

JUMP, 2005. http://www.jump-project.org/ (accessed 2005-02-09).

Laser-Scan, 2005. http://www.laser-scan.com/ (accessed 2005-02-09).

Lehto, L., and Kilpeläinen, T., 2000. Real-Time Generalisation of Geodata in the Web. *International Archives of Photogrammetry and Remote Sensing*, Vol. XXXIII, Part B4, Amsterdam, pp. 559-566.

Lehto, L. and Kilpeläinen, T., 2001a. Real-time Generalisation of XML-encoded Spatial Data on the WEB. *Proceedings of GIS Research UK*, 9th Annual Conference GISRUK 2001, University of Glamorgan, Wales, pp. 182-184.

Lehto, L. and Kilpeläinen, T., 2001b. Generalizing XML-encoded Spatial Data on the Web. *Proceedings of the 20th International Cartographic Conference*, August 6-10, 2001, Beijing, China, Volume 4, pp. 2390-2396.

Lehto, L., 2003. *GiMoDig system architecture*, Available at http://gimodig.fgi.fi/deliverables.php (accessed 2003-09-10).

OGC, 2005. Open Geospatial Consortium – Simple Features Specification for SQL. (available at http://www.opengeospatial.org/docs/99-049.pdf, 2005-02-11)

Ruas, A., and Plazanet, C., 1996. Strategies for Automated Generalization. *Proceedings of the 7th Spatial Data Handling Symposium*, Delft, the Netherlands, pp. 319-336.

Sarjakoski, T., and Kilpeläinen, T., 1999. Holistic Cartographic Generalization by Least Squares Adjustment for Large Data Sets*, Proceedings of the 19th International Cartographic Conference*, Ottawa, pp. 1091-1098.

Sarjakoski, T., and Lehto, L., 2003, Mobile Map Services Based on an Open System Architecture. *Proceedings of the 21st International Cartographic Conference*, 10 - 16 August 2003, Durban, South Africa, pp.1107-1113.

Sarjakoski, T., Sester, M., Sarjakoski, L. T., Harrie, L., Hampe, M., Lehto, L., and Koivula, T., 2005. Web Generalisation Service in GiMoDig – towards a standardised service for real-time generalisation. Submitted to *Agile conference*, Estoril, Portugal.

Shewchuk, J. R., 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In: M. C. Lin and D. Manocha, (eds), *Applied Computational Geometry: Towards Geometric Engineering*, vol 1148. Lecture Notes in Computer Science, pp. 203-222. Springer-Verlag.

Sester, M., 2000. Generalization Based on Least Squares Adjustment, *International Archives of Photogrammetry and Remote Sensing,* Vol. XXXIII, Part B4, Amsterdam, pp. 931-938.

Sester, M., Sarjakoski L. T., Harrie, L., Hampe, M., Koivula, T., Sarjakoski, T., Lehto, L., Elias, B., Nivala, A.-M. and Stigmar, H., 2004. Real-time generalisation and multiple representation in the GiMoDig mobile service. GiMoDig-project, IST-2000-30090, Deliverables D7.1.1, D7.2.1 and D7.3.1, Public EC report, 151 p. An electronic version available at http://gimodig.fgi.fi/deliverables.php

Stigmar, H., 2005. Matching Route Data and Topographic Data in a Real-time Environment. Submitted to ScanGIS, 2005, Stockholm.

van Kreveld, M., 2001, Smooth Generalization for Continuous Zooming. *Proceedings of the 20th International Cartographic Conference*, 6 - 10 August 2001, Beijing, China , pp.2178-2185.

Vivid Solutions, 2005. *JTS Topology Suite*, http://www.vividsolutions.com/jts/jtshome.htm (accessed 2005-02-09).

Ware, J. M., and Jones, C. B., 1998. Conflict Reduction in Map Generalization Using Iterative Improvement. *GeoInformatica,* Vol. 2, No. 4, pp. 383-407.

WFS, 2005. *Web Feature Service Implementation Specification*. http://www.opengis.org/docs/02-058.pdf (accessed 2004-05-28).

WMS, 2005. *Web Map Service Implementation Specification*. http://www.opengis.org/docs/01-068r2.pdf (accessed 2004-05-28).

Zhang, Q, and Harrie, L., 2005. Real-Time Map Labelling for Mobile Applications. *Computers, Environment and Urban Systems*, accepted.